

iniciando javascript

Conhecendo o Javascript
Comentários, Strings e Numbers,
Fazendo cálculos com JavaScript,
operadores relacionais e comparativos
Condicionais,
objetos e
Arays.

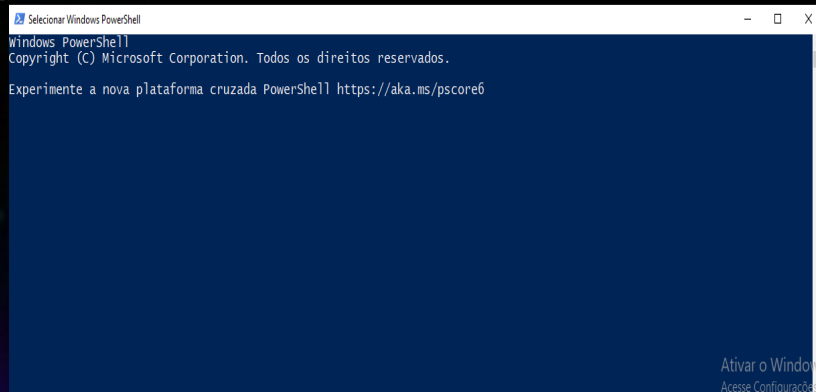
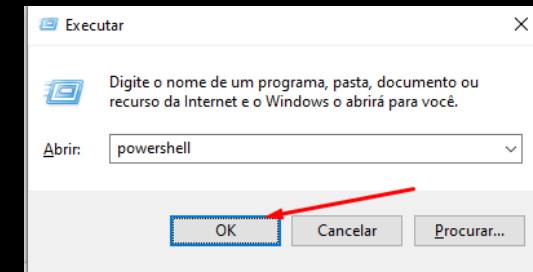
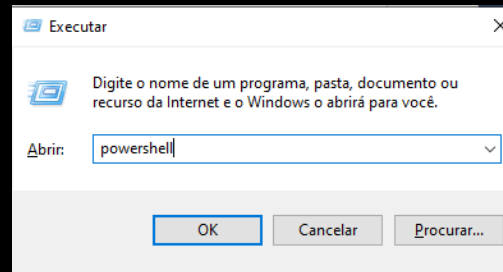
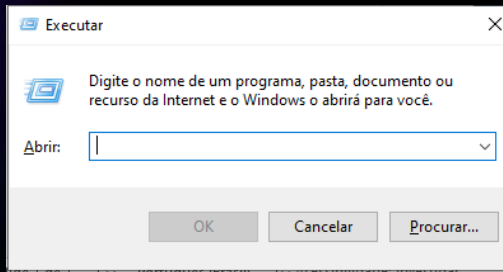


PowerShell



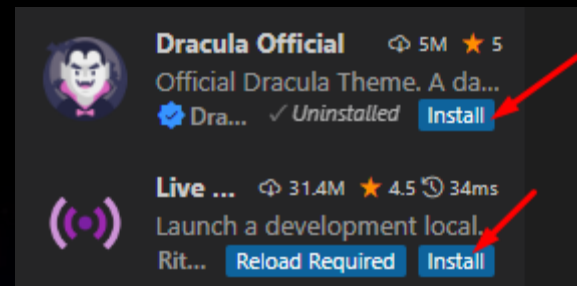
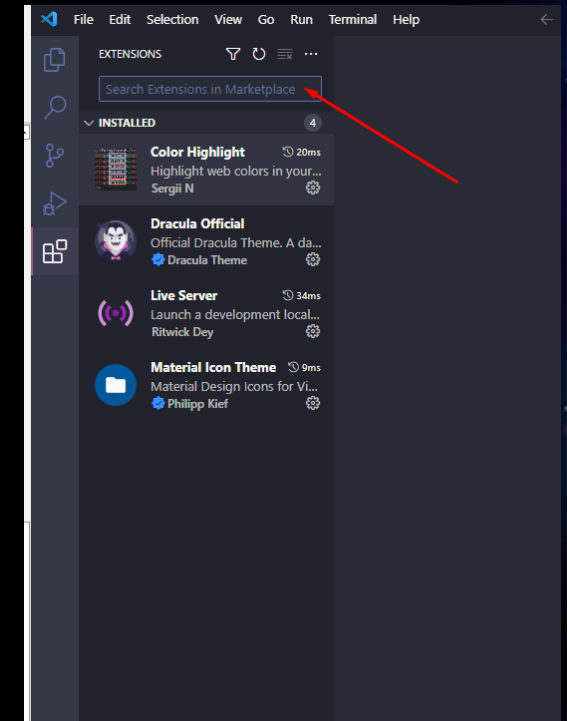
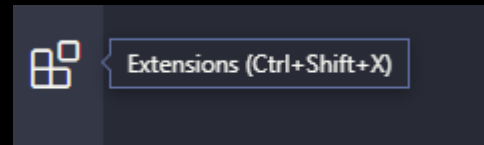
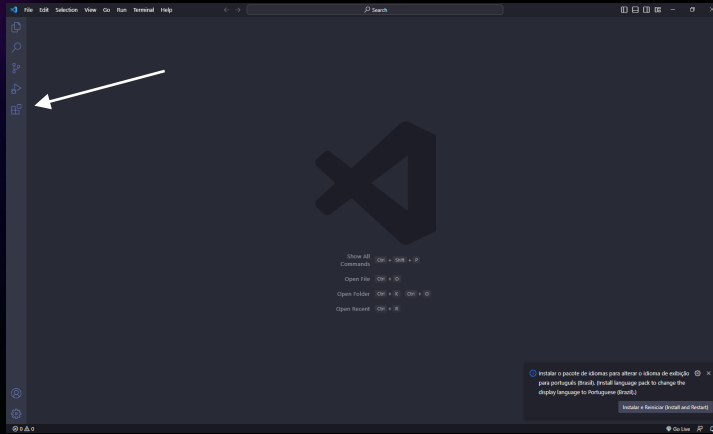
que a força esteja com você.
Prof. fernando Lucas

1 – vamos utilizar o powershell para abrir o vscode e criar uma pasta que será onde guardaremos nosso projeto.

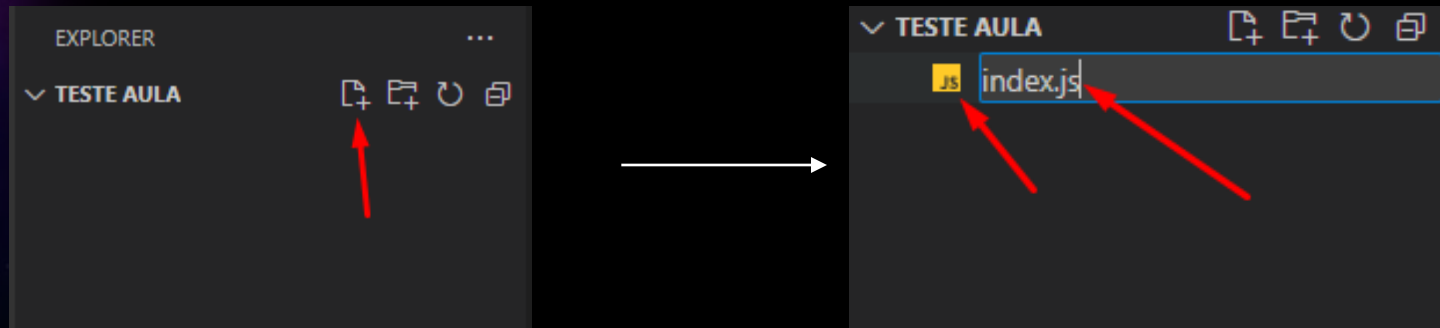


Comandos –
dir – lista os arquivos
Cd 'nome da pasta' – navega nas pastas
Mkdir – cria uma pasta nova
Code . – abre o vscode na pasta que você esta.

2 – Com vscode aberto vamos adicionar nossas extensões.



3 – criando arquivo JavaScript no vscode.



Escolha um nome que por padrão é index, e repare que o que você coloca depois do ‘.’ é que define o tipo do arquivo.

Poderia ser:

index.html

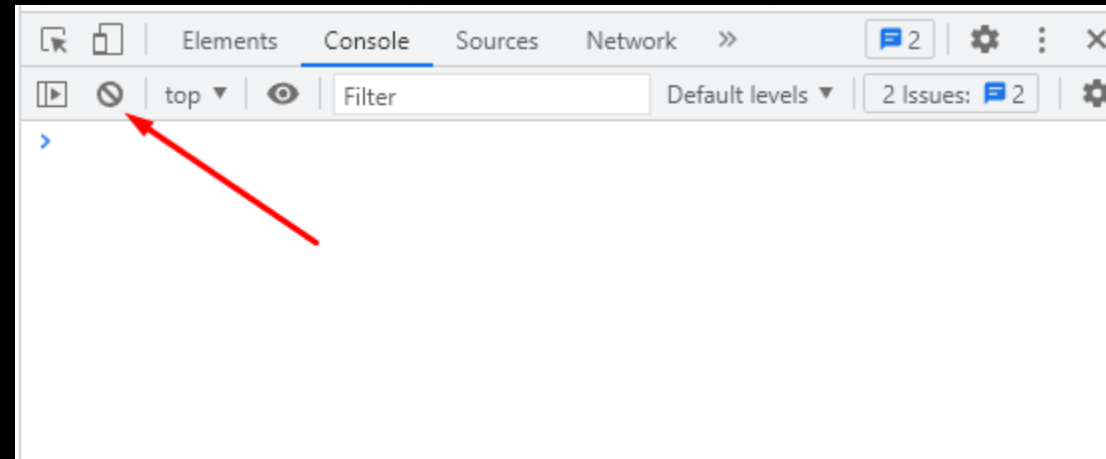
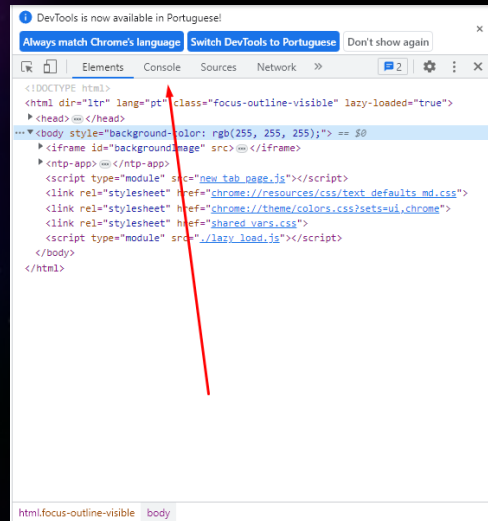
index.css

index.js

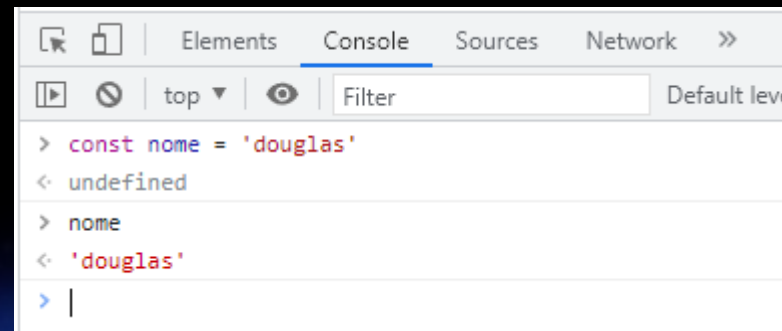
index.pkg

Entre outras 1000 linguagens que existem.

4 – Acesse o console do chrome apertando f12



Crie uma constante e execute ela em Javascript no motor do chrome.



5 – criando uma constante e mostrando ela no console.

```
JS index.js > ...  
1  const nome = 'juliana';  
2  console.log(nome);  
3
```

Console.log(nome)

Ctrl j

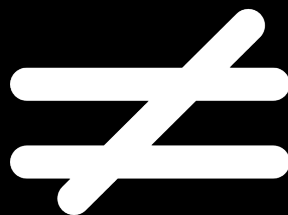
Para abrir o console

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  
PS D:\teste aula> node index.js  
juliana  
PS D:\teste aula>
```




6 – Podemos escrever **comentários**, que são palavras que não serão interpretadas como códigos ou comandos, serve principalmente para informar algo.

```
// Criar um programa que calcula a média  
// das notas entre os alunos e envia  
// mensagem do cálculo da média.
```

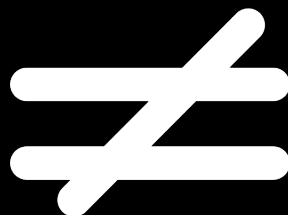


```
const nome = "Mayk"  
const nome2 = 'Diego'
```

7 – criando e entendendo a diferença entre uma variável do tipo 'string' e 'template string'.

```
const nome2 = 'Diego ${nome}'
```

– dessa forma tudo dentro da variável será reconhecido como um texto '-----'.



```
const nome3 = `Valeska e ${nome}`
```

– e utilizando **`${nome}`** podemos chamar outra variável dentro dela.

Crie uma de cada variável e imprima elas com o comando `console.log()`

8 – agora vamos criar uma variável do tipo `number`.

```
const notaAluno01 = 9.8
```

Caso você queira saber o tipo de variável que você criou existe um comando para isso -

```
console.log(typeof notaAluno01)
```

Mostra
No console

Mostra
o tipo

o que
vai ser
Testado

imprima o comando `console.log(typeof ^^^^)`



9 – Criando uma calculadora de media dos alunos.

Crie três variáveis do tipo `number` Para guardar as notas.

crie uma variável para guardar o calculo de media.

Mostre tudo isso no console !



Criando uma calculadora de media dos alunos.

```
const notaAluno01 = 9.8  
const notaAluno02 = 10  
const notaAluno03 = 2
```

Crie três variáveis do tipo **number** Para guardar as notas.

```
const media = (notaAluno01 + notaAluno02 + notaAluno03) / 3
```

crie uma variável para guardar o calculo de media.

```
console.log(media)
```

Mostre tudo isso no console !

```
node index.js
```

Agora vamos pro upgrade



utilize uma estrutura condicional para parabenizar a turma Com uma mensagem no console **se** a **media** for maior que 5 **e se não for já sabe.**

dica

```
IF (VARIABLE > 5){  
  CONSOLE.LOG ( ----- ) }  
  
else{  
  Console.log ( -----) }
```

desafio

utilize uma template string para mostrar a variável media dentro da mensagem.

Agora vamos pro upgrade



utilize uma estrutura condicional para parabenizar a turma Com uma mensagem no console se a media for maior que 5 e se não for já sabe.

desafio

utilize uma template string para mostrar a variável media dentro da mensagem.

```
if (media > 5) {  
    console.log(`A nota foi de ${media}. Parabéns`)  
} else {  
    console.log('A média é menor que 5')  
}
```

Agora vamos pro upgrade

Na casinha !

OPERADORES DE COMPARAÇÃO

>	Maior
<	Menor
>=	Maior igual a
<=	Menor igual a
==	Igual a
===	Igual e do mesmo tipo
!=	Diferente de
!==	Diferente, inclusive do tipo

// DESAFIO 1

```
// se sim, deixar entrar, se não, bloquear a entrada  
// se a pessoa tiver 17 anos  
// avisar para voltar quando fizer 18 anos
```

// DESAFIO 2

```
// dar bonificação de 1.000  
// se o vendedor possuir mais que 100 pontos
```

que a força esteja com você.
Prof. fernando Lucas

operadores lógicos

OPERADORES DE LÓGICOS

```
&& "E" As duas condições devem ser verdadeiras  
| para que a condição final seja verdadeira.  
|| "OU" Uma das condições deve ser verdadeira  
| para que a condição final seja verdadeira.  
! "NÃO" Nega uma condição
```

Melhore o seu código

```
// DESAFIO 1  
// se sim, deixar entrar, se não, bloquear a entrada  
// se a pessoa tiver 17 anos  
// avisar para voltar quando fizer 18 anos
```

objetos :3

imagine, seu cadastro na escola é encontrado no sistema pelo seu nome.

uma aluna chamada **lais**
no sistema da escola **tem varias informações como:**

Endereço

Telefone

Nota

Sala

Turno

agora imagine se o sistema tivesse que criar uma variável

Para cada informação da **lais**.

vamos converter essa situação pro **JavaScript**.

objetos :3

Endereço

Telefone

Nota

Sala

Turno

agora imagine se o sistema tivesse que criar uma variável

Para cada informação da **lais**.

vamos converter essa situação pro **JavaScript**.

```
const nome = lais;  
const enderecoDaLais = "rua kennedy n4"  
const notaFinal = 9.8  
const salaDaLais = 5  
const turnoDaLais = "manha"
```

objetos :3

```
const nome = lais;  
const enderecoDaLais = "rua kennedy n4"  
const notaFinal = 9.8  
const salaDaLais = 5  
const turnoDaLais = "manha"
```

Bom, percebemos que lotamos nosso programa de variáveis e isso certamente não é bom.
Mas solucionamos nosso problema é claro.

E se tivesse mais alunos na escola?

Já imaginou quantas variáveis teríamos que criar ?

```
const nome = lais;  
const enderecoDaLais = "rua kennedy n4"  
const notaFinal = 9.8  
const salaDaLais = 5  
const turnoDaLais = "manha"  
  
const nome = lais;  
const enderecoDaLais = "rua kennedy n4"  
const notaFinal = 9.8  
const salaDaLais = 5  
const turnoDaLais = "manha"  
  
const nome = lais;  
const enderecoDaLais = "rua kennedy n4"  
const notaFinal = 9.8  
const salaDaLais = 5  
const turnoDaLais = "manha"  
  
const nome = lais;  
const enderecoDaLais = "rua kennedy n4"  
const notaFinal = 9.8  
const salaDaLais = 5  
const turnoDaLais = "manha"  
  
const nome = lais;  
const enderecoDaLais = "rua kennedy n4"  
const notaFinal = 9.8  
const salaDaLais = 5  
const turnoDaLais = "manha"
```

objetos :3

Certamente isso não seria **good** pra gente.

Pra isso nos usamos um método chamado de **objeto**.

E se pudéssemos guardar todos os dados da lais dentro de uma variável só? E melhor e se pudéssemos chamar cada informação somente chamando a variável lais.

Antes nos pensávamos em variáveis como uma pequena caixa.

Agora vamos pensar em objetos como uma pequena cômoda cheia de gavetas.

Estrutura do objeto em JavaScript:

```
1  const lais = {  
2      |   endereco: "rua kennedy n4",  
3      |   notaFinal: 9.8,  
4      |   sala: 5,  
5      |   turno: "manha"  
6  }  
7  console.log (lais.notaFinal)
```

objetos :3

Agora vai ficar **good**, posso chamar somente
A informação que eu quero.

Chame no console somente a informação que você quer imprimir.

```
1  const lais = {  
2      endereco: "rua kennedy n4",  
3      notaFinal: 9.8,  
4      sala: 5,  
5      turno: "manha"  
6  }  
7  console.log (lais.endereco)
```

```
1  const lais = {  
2      endereco: "rua kennedy n4",  
3      notaFinal: 9.8,  
4      sala: 5,  
5      turno: "manha"  
6  }  
7  console.log (lais.notaFinal)
```

Tudo que está dentro do objeto chamamos de
Propriedades.

```
1  const lais = {  
2      endereco: "rua kennedy n4",  
3      notaFinal: 9.8,  
4      sala: 5,  
5      turno: "manha"  
6  }  
7  console.log (lais.endereco)
```

usamos **'{}'** para iniciar o objeto
usamos a **','** para separar as propriedades
usamos o **'.'** para acessar as propriedades

objetos :3

uma curiosidade, estamos usando o comando `console.log ()`

E esse `danado` é literalmente um objeto.

`console.log ()`

↓
objeto javascript

↙
. Para chamar uma
Propriedade.

↓
A propriedade
escolhida.

arrays o/

Ainda no sentido de diminuir nosso numero de variáveis temos os

arrays

que são uma forma de armazenar todos os objetos em uma variável.
Declarando dessa forma....

arrays o/

vamos criar um programa que guarda as informações dos alunos de uma determinada escola.

```
const escola = []
```

Primeiro criamos o **array** e damos um nome a ele,
São os **[]** que vão fazer com que nossa variável se torne um **array**.

```
const escola = [{}, {}, {}, {}]
```

Dentro de cada **{ }** haverá um objeto com suas
Propriedades.

arrays o/

Agora temos que entender que cada objeto ganhou uma **posição** no nosso **array**.

Começando a contar do **numero 0**.

```
const escola = [ {}, {}, {}, {} ]
```



```
const escola = [
```

```
{  
  nome: "lais",  
  endereco: "rua kennedy n4",  
  notaFinal: 9.8,  
  sala: 5,  
  turno: "manha"},
```

0

```
{  
  nome: "pedro teles",  
  endereco: "rua amanha tem copa",  
  notaFinal: 9.8,  
  sala: 5,  
  turno: "manha"},
```

1

```
{  
  nome: "savio",  
  endereco: "rua kenya west",  
  notaFinal: 9.8,  
  sala: 5,  
  turno: "manha"},  
]
```

2

arrays o/

Agora com o nosso comando god dos testes o `console.log ()`.
vamos a partir da posição do **array** imprimir as informações do Pedro.

```
const escola = [
```

```
{  
  nome: "lais",  
  endereco: "rua kennedy n4",  
  notaFinal: 9.8,  
  sala: 5,  
  turno: "manha"},
```

0

```
{  
  nome: "pedro teles",  
  endereco: "rua amanha tem copa",  
  notaFinal: 9.8,  
  sala: 5,  
  turno: "manha"},
```

1

```
{  
  nome: "savio",  
  endereco: "rua keny west ",  
  notaFinal: 9.8,  
  sala: 5,  
  turno: "manha"},
```

2

```
console.log(escola[1].endereco, escola[1].nome)
```

E assim conseguimos chamar as informações que queremos.

arrays o/

Agora é sua vez !!!!!

Crie um **array** com as seguintes diretrizes:

1 – 5 posições.

2 – cada posição um **objeto** com 3 propriedades.

3 – imprima na tela tudo que estiver na terceira posição do **array**.

funções o/

Pensando em reaproveitar nossos códigos

função

Ela permite que possamos reaproveitar um bloco de código varias vezes, sem precisar escreve-lo novamente.

funções o/

Ela possui dois momentos

- 1 – declarar
- 2 - chamar

funções o/

declarando

```
function soma (a,b){}
```

comando

o nome

os parâmetros
que serão recebidos
Dentro da função
E executados.

os comandos que
ela deve executar.

return

o Retorno
Da função.

funções o/

chamando

```
soma (5,6)
```

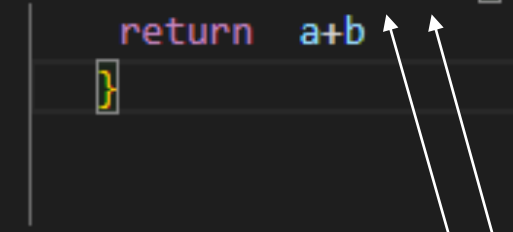
↓
nome

↓
os parâmetros
que serão enviados para dentro da função.

funções o/

Exemplo

```
function soma (a,b) {  
  return a+b  
}  
  
console.log(soma (5,6))
```

Two white arrows originate from the parameters '5' and '6' in the function call 'soma (5,6)' and point to the parameters 'a' and 'b' in the function definition 'function soma (a,b)'. This illustrates how the arguments provided in the call are passed to the function's parameters.

OS parâmetros da nossa função são a e b (a,b), quando nos chamamos nossa função precisamos entregar os parâmetros para serem substituídos dentro dela, neste caso é o (5,6).

TRy ai man !

Estrutura de Repetição o/

Pensando em repetir comandos

Estrutura de Repetição

- For -

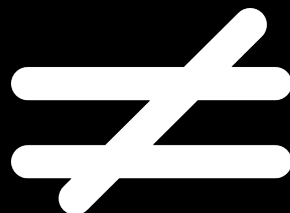
Ela permite que possamos repetir o mesmo código varias vezes.

Nova configuração de variável desbloqueada

Let

Diferente da configuração `const`, o `let` consegue
Alterar seu valor a qualquer momento do programa.

```
const qualquerNome = 5
```



```
let qualquerNome = 5
```

Exemplo

```
for (let i = 0 ; i < 9; i++){  
  console.log(i)  
}
```

o comando
i++
É a mesma coisa de i
+ 1.

o **for** funciona como um contador, enquanto a condição não for verdadeira ele vai continuar rodando o código.

E sempre que ele executar o código ele vai alterar o valor da variável de controle, no nosso caso (**i**).

Então sempre que o código é executado é adicionado +1 a variável let (**i**).

TRy ai man !